User-Oriented FEC Decoding Scheme for Wireless Sensor Networks with Star Topologies

Fan Yang† and Xi Zhang‡

†School of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, E-mail: {*yangf_byr@yahoo.cn*}

‡Networking and Information Systems Laboratory, Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA, E-mail: {*xizhang@ece.tamu.edu*}

***Abstract*—In recent years, wireless sensor networks (WSNs) have attracted a great deal of research attention from both academia and industry. In most cases, WSNs employ the adaptive forward error correction (FEC) coding schemes dynamically adapting to the channel, while paying less attention to user's requirements. Sometimes, the decoding delay and the decoding performance of WSNs are more crucial than the power consumption for users. We propose an FEC decoder for the trade-off among decoding delay, complexity, and performance according to user's needs. The proposed decoder designed for short packet transmission is a two-stage hybrid decoder (TSHD) including a serial min-sum decoder and an adaptive list decoder. For hard decision decoding, our simulation results show that the proposed TSHD scheme outperforms the existing schemes with less average complexity.**

***Index Terms*—M-algorithm, serial min-sum decoder, hybrid decoder, real-time, low-density parity-check codes.**

I. Introduction

Wireless sensor networks (WSNs) are widely used in industry, medical assistance, video surveillance, etc. In WSNs, a large number of small sensor nodes are deployed surround the sensed object to collect data for users [1]. The most important challenge is to design sensors efficiently to minimize the wireless sensor resource. In order to lower the decoding latency and reduce the transmitting power, WSNs tend to include an forward error correction (FEC) coding to avoid retransmissions [2]. Since the decoding processing may consume much power, the FEC coding is often applied to a star-WSN with sensed data from sensor nodes routed via a central node [3]. The sensor nodes of the star-WSN are energy-constrained, while the central node often has a relatively larger energy resource.

Because a large amount of data need to be processed in the central node, the reduction for its decoding complexity without performance loss is still necessary. Furthermore, wireless environment conditions can be time-varying, the adaptive algorithms are favorable for WSNs scenarios. Moreover, these algorithms should be aware of user's requirements. We take fire disaster monitoring for instance. Usually, reduction for the power consumption is the primary task. Once the fire disaster happens, the data of the real situation should be transmitted to the monitor with low latency and high data precision. Thus, the trade-off among power, latency, and performance is important to user's needs. Empirically, more power should be consumed for the sake of the low latency and the performance improvement. Due to hardware limitations, we analyze the decoding complexity instead of measuring the power consumption. In [4], convolutional codes were first addressed for WSNs. Later on, algebraic codes such as BCH codes and Reed-Solomon (RS) codes were proposed in [5], and LDPC codes were considered due to its remarkable performance [6].

Convolutional codes are decoded on a trellis using either Viterbi decoding, MAP decoding, or sequential decoding, while M-algorithm decoding remains unexplored for convolutional codes applying to sensor network. Compared with the sequential decoding, the M-algorithm decoding can be implemented with constant storage and complexity for long constraint codes [7]. Moreover, the authors of [8] proved that a systematic feed-forward convolutional code is better than a non-systematic convolutional code when using an M-algorithm decoder, the former allows quick recovery of a lost correct path. They also indicated that M-algorithm with a systematic feed-forward encoder is 2-4 times more efficient than the Viterbi algorithm in terms of survivor paths. On the other hand, BCH and RS codes are decoded using either syndrome decoding, algebraic decoding or chase algorithm decoding.

Usually, the transmitting packet size is often changed according to user's requirements or specified channel qualities, which can be flexibly adjustable using convolutional codes. LDPC codes decoded based on a bipartite graph use either belief propagation (BP) decoding or simplified version of BP called min-sum decoding. The min-sum decoding requires real additions only, and is suitable to hardware implementation with quantized received values.

As authors of [4] indicated that current commercial radio transceivers are not ideal for WSNs' applications, the innovative solutions in transceiver design are required to transmit short packets effectively. In recent years, the researchers are interested in convolutional-type codes, such as low density parity check convolutional codes (LDPCCCs), convolutional self-orthogonal codes (CSOCs) and convolutional self-doubly-orthogonal codes (CSO2Cs) [9], [10]. In [11], [12], repeat-accumulate codes have been used in single-hop or multiple

This work was supported in part by the U.S. National Science Foundation under Grant CNS-205726 and the U.S. National Science Foundation CAREER Award under Grant ECCS-0348694.

978-1-4673-5239-0/13/$31.00 ©2013 IEEE

relay for WSNs. All of the above schemes are decoded by BP algorithm or its simplified versions, however, the memory requirement of these algorithm is very large due to iterative decoding. Moreover, the use of iterative decoding technique is not well suitable to delay-intolerant scenarios, and the ensuring for the independency of the extrinsic information needs transmission long packets.

Traditionally, LDPC decoders and trellis decoders are performed separately based on their own code structures, since convolutional codes decoded by BP algorithms directly always results in poor performance, this is mainly due to a large number of short cycles (especially 4-cycles) in its parity check matrix. In our previous work [13], we proposed a method to design systematic feed-forward convolutional codes with no 4-cycles (SFCC-N4) which has a sparse parity check matrix and is suitable for BP decoding. In this paper, we try to combine the LDPC decoder and the trellis decoder together for application in star-WSNs. In our proposed scheme, the SFCC-N4 code is used as the encoder and its associated decoder is a two-stage hybrid decoder (TSHD) with a modified serial min-sum decoder and an adaptive list decoder. The TSHD has several merits such as adaptive optimization, low average computation without performance loss, suitable for short packet transmissions. As we will show, the TSHD has a good trade-off among complexity, decoding delay, and performance.

The rest of the paper is organized as follows. In Section II, the TSHD is introduced by presenting the modified serial min-sum decoder and the adaptive list decoder, respectively. In Section III, we analyze the decoding complexity of TSHD by comparing it with conventional single M-algorithm decoder (SMAD). Then simulation results are provided, and the trade-off for adaptive optimization are shown in Section IV. Finally, Section V presents the conclusion.

II. The Two-Stage Hybrid Decoder

Before describing TSHD, it should be emphasized that the concept *adaptive optimization* is somewhat different from *adaptive error control*. Fig. 1 shows the transmission from sensor node to central node in a single-hop. The adaptive error control module changes the parameters according to channel conditions. For example, the encoder can increase the code rate or choose less powerful codes when the channel is good, and vice versa. The adaptive optimization module varies parameters followed by user's requirements or the control signal. If user wants to shorten the decoding delay while maintain the performance, then the adaptive optimization module can increase code rate or apply more powerful codes for the encoder, and can increase the list size for the decoder. That means no matter how good the channel condition is, more power should be consumed for the sake of low latency. Clearly, the priority of adaptive optimization is higher than that of adaptive error control.

In the following, the rate-1/2 binary SFCC-N4 code is considered over the additive white Gaussian noise (AWGN) channel using BPSK modulation. The codeword $\mathbf{c} = \{c_n\}$ is mapped into a bipolar transmitted sequence $\mathbf{x} = \{x_n\}$, for

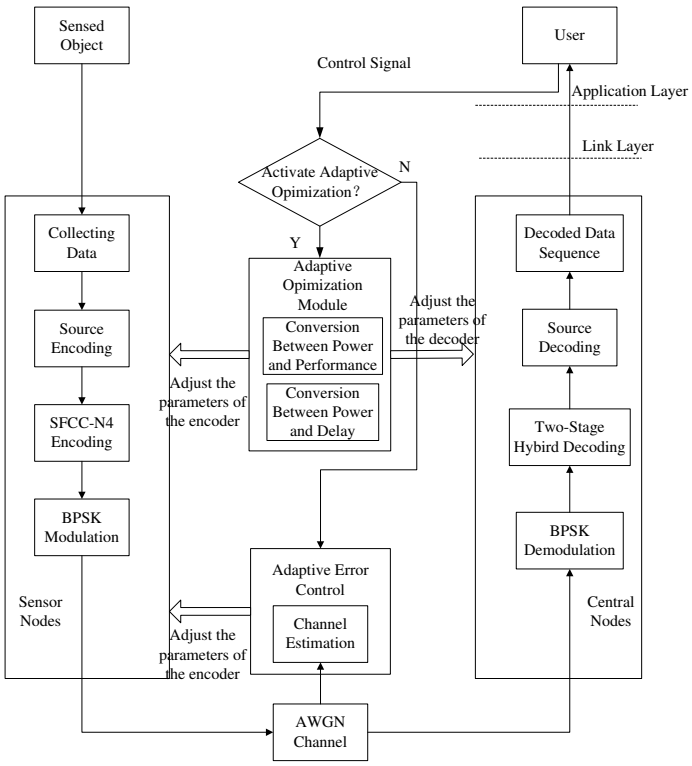


Fig. 1. User-oriented adaptive optimization model.

$1 \le n \le 2N$. Then the received vector $\mathbf{y} = \mathbf{x} + \mathbf{w}$, where $\mathbf{w} = \{w_n\}$, and w_n is a random variable with zero mean and variance σ^2. Let $\hat{\mathbf{x}} = \{\hat{x}_n\}$ be the output binary sequence of the modified serial min-sum decoder, and $\hat{x}_{[2t-1,2t]} = (\hat{x}_{2t-1}, \hat{x}_{2t})$ is the estimated symbol generated by the modified serial min-sum decoder at time t with a systematic bit $\hat{x}_{2t-1}$ and a parity check bit $\hat{x}_{2t}$, such that $1 \le t \le N$.

In TSHD, the first stage employs low complexity modified serial min-sum decoder to explore the received vector, and the second stage applies the powerful M-algorithm to correct rest errors. In the modified serial min-sum decoder, the variable nodes are updated one-by-one in sequence and a check node is activated whenever its message is demanded by a neighboring variable node. At the first stage, the modified serial min-sum decoder generates a binary estimated coded sequence, which might not be a valid codeword. Since there is a one-to-one correspondence between a valid codeword and a continuous path in trellis, the adaptive list decoder gradually forms a continuous single path by verifying the generated symbol according to the trellis structure. Whenever the path departs from a valid codeword at trellis depth t (or time t), the adaptive list decoder applies M-algorithm to re-decode the erroneous segment from time t' to t by extending the single path to M_p survivor paths, where M_p is the list size of the M-algorithm decoder. The recovered partial path corresponding to the segment is the survivor path with the largest cumulated metric among the M_p survivor paths. Then the verification process continues from time t, and the M_p paths return to a single path again. The remaining process is the same as above

until reaching the end of the trellis.

A. The Modified Serial Min-Sum Decoder

A regular binary $(2N, J, 2J)$ LDPC code with J "1"s in each column and $2J$ "1"s in each row can be described by a sparse parity check matrix $\mathbf{H}$ of size $M \times 2N$. The min-sum algorithm is composed of two phases of message passing, i.e., the variable-to-check node message passing $L(q_{nm})$ and the check-to-variable node message passing $L(r_{mn})$. $L(q_n)$ denotes the *a posteriori* log-likelihood ratio (LLR) of variable n. $L(p_n)$ represents the *a priori* LLR of variable n. Moreover, $N(m) = \{n|h_{mn} = 1\}$ denotes the set of variable nodes connecting to check node m, and $M(n) = \{m|h_{mn} = 1\}$ denotes the set of check nodes connecting to variable node n. $N(m)\backslash\{n\}$ represents the set $N(m)$ other than variable node n, similarly, $M(n)\backslash\{m\}$ represents the set $M(n)$ other than check node m.

Compared with the parallel min-sum algorithm, the convergence speed of the serial min-sum algorithm is faster with less storage requirement. The modified serial min-sum decoder is a serial algorithm which updates the variable nodes one-by-one in sequence and generates a continuous output after L time units. Moreover, the modified serial min-sum decoder can be implemented for real-time structure since it executes only one iteration.

Let α_{nm} and β_{nm} be the sign and the magnitude of $L(q_{nm})$, respectively. According to the parallel min-sum decoder [14], the conventional serial min-sum decoder needs to serially calculate $\min_{\tilde{n}\in N(m)\backslash\{n\}} \beta_{\tilde{n}m}$ for each variable node n in check m, requiring $2NJ(2J + \lceil \log_2 2J \rceil - 2)$ additions for this operation in one iteration, however, it is rather complex for large J values. Note that the minimum values $\min_{\tilde{n}\in N(m)\backslash\{n\}} \beta_{\tilde{n}m}$ associated with each check m can be determined by identifying the two minimum values corresponding to this check [14]. Suppose γ_1 and γ_2 are the two minimum values corresponding to set $n \in N(m)$ before updating β_{nm}, such that $\gamma_1 < \gamma_2$. Once β_{nm} is updated, let γ_u be the value of the updated β_{nm}, then γ_1 and γ_2 may no longer be the two minimum values of this set. At the nth time unit, the two minimum values of check m can be determined by comparing three values (γ_1, γ_2 and γ_u), i.e., the two minimum values among the three are regarded as the new γ_1 and γ_2. The modified serial min-sum decoder can be summarized as follows:

Initialization: For each m, n, set $L(p_n) = y_n$, $L(q_{nm}) = L(p_n)$. For each check m, compute γ_1 and γ_2.

One-iteration processing: Update $L(q_{nm})$ in four steps according to $n = 1, 2, ..., 2N$.

1) *Step 1*: For the given n and $m \in M(n)$, compute

$$L(r_{mn}) = \left(\prod_{\tilde{n}\in N(m)\backslash\{n\}} \alpha_{\tilde{n}m} \right) \min_{\tilde{n}\in N(m)\backslash\{n\}} \beta_{\tilde{n}m} \quad (1)$$

The calculation for $\min_{\tilde{n}\in N(m)\backslash\{n\}} \beta_{\tilde{n}m}$ is replaced by using γ_1 or γ_2.

2) *Step 2*: For the given n and $m \in M(n)$, compute

$$L(q_n) = L(p_n) + \sum_{m\in M(n)} L(r_{mn}) \quad (2)$$

$$L(q_{nm}) = L(q_n) - L(r_{mn}) \quad (3)$$

Then update γ_1 and γ_2 according to the rule: If $\gamma_u < \gamma_1$, set $\gamma_2 = \gamma_1$ and $\gamma_1 = \gamma_u$; If $\gamma_1 \le \gamma_u \le \gamma_2$, maintain γ_1 and set $\gamma_2 = \gamma_u$; Otherwise, $\gamma_2 < \gamma_u$, γ_1 and γ_2 are not changed.

3) *Step 3*: If $n < 2N$, the estimated bit $\hat{x}_n$ is determined by $L(q_n)$ such that $\hat{x}_n = 1$ if $L(q_n) < 0$, and $\hat{x}_n = 0$ if $L(q_n) > 0$. Then $n = n + 1$, go to step 1; Otherwise, go to step 4.

4) *Step 4*: Create $\hat{\mathbf{x}} = \{\hat{x}_n\}$. If $\mathbf{H}\hat{\mathbf{x}}^T = \mathbf{0}$, then $\hat{\mathbf{x}}$ is considered as a valid decoded codeword and the processing ends; Otherwise, a failure is declared, apply adaptive list decoder for re-decoding.

B. The Adaptive List Decoder

Considering a single-stage trellis from time $t-1$ to t, s_{t-1} and s_t denote the two states connected by branch b_t. $\bar{x}_{[2t-1,2t]} = (\bar{x}_{2t-1}, \bar{x}_{2t})$ represents the branch symbol corresponding to b_t. The verification for a generated symbol and a branch symbol is to assume they have the same systematic bit, then check whether the parity check bit equal or not. In other words, assume $\bar{x}_{2t-1} = \hat{x}_{2t-1}$, check if $\bar{x}_{2t}$ and $\hat{x}_{2t}$ are equal.

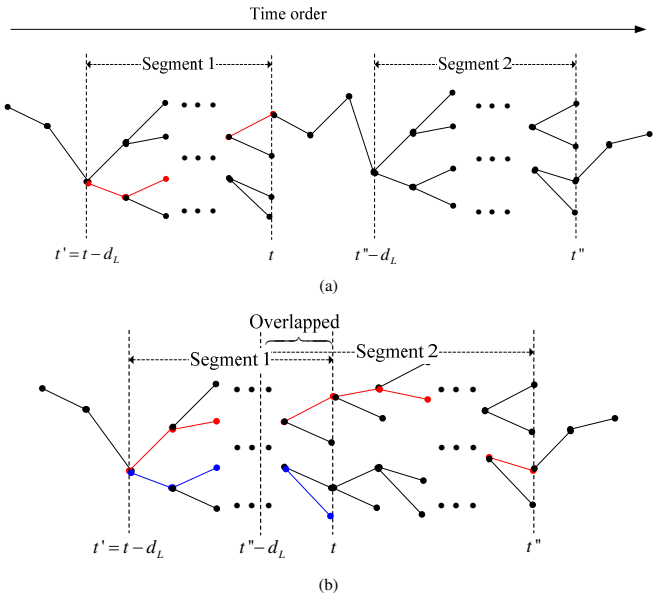


Fig. 2. An example for adaptive list decoder procedure ($M_p = 2$). (a) Two segments are separated satisfying $t < t'' - d_L$. (b) Two segments are overlapped such that $t \ge t'' - d_L$.

The list size of adaptive list decoder has two levels corresponding to a single path and M_p survivor paths. When the first symbol is generated by the modified serial min-sum decoder, adaptive list decoder starts the verification process from the zero state by comparing it with branch symbol $\bar{x}_{[1,2]}$. Assuming that $S_0^{t-1} = (0, s_1, s_2, .., s_{t-1})$ is a continuous single path formed from time 0 to $t-1$. Considering the tth time unit, if $\hat{x}_{[2t-1,2t]} = \bar{x}_{[2t-1,2t]}$, or $\hat{x}_{2t} = \bar{x}_{2t}$, the path extends to S_0^t by adding the state s_t to the end of S_0^{t-1}; Otherwise, the mismatching of the two symbols indicates that

the path is broken at trellis depth t, then adaptive list decoder uses M-algorithm to re-decode the received symbols in the segment from time t' to t, the single path $S_0^{t'}$ extends to M_p survivor paths from state $s_{t'}$, where $t' < t$ in time order. Let d_L be the backtracking depth such that $d_L = t - t'$. After that, a partial path $S_{t'}^t = (s_{t'}, s_{t'+1}, ..., s_t)$ of S_0^t is recovered by M-algorithm, which is the survivor path with the largest metric among M_p survivor paths (the red line in Fig. 2 (a)). The verification process goes on until S_0^N is formed. By the way, when the path is broken twice (or more) within a time interval less than d_L, i.e., the path is broken at time t and later at time t'' such that $t'' - t \leq d_L$, where $t' < t < t''$ in time order (see Fig. 2 (b)). Our method is to continue M-algorithm for decoding the received symbols from time t to t'' to recover $S_{t'}^{t''}$, and $S_{t'}^t$ has been updated (In Fig. 2 (b), the blue line corresponds to $S_{t'}^t$, and the red line corresponds to $S_{t'}^{t''}$). Thus, the trellis stages from $t'' - d_L$ to t has decoded only once, there is no repetition for trellis decoding and no extra computation is required.

C. Implementation of Two-Stage Hybrid Decoder

Although TSHD has two steps of decoding, as mentioned above, it still can be implemented as a real-time decoder with fixed decoding delay. Let L be the constraint length of the SFCC-N4 code. Each time two channel outputs enter the decoder, and it takes L initial time units for the modified serial min-sum decoder to generate a symbol, then adaptive list decoder starts the verification from all zero state through trellis. After L time units, the modified serial min-sum decoder and the adaptive list decoder can be processed in parallel. The decoding delay of adaptive list decoder depends on the backtracking depth d_L, and accordingly, the decoding delay for TSHD is $L + d_L$ time units. In addition, the storage requirement for TSHD is a little more than SMAD because of the additional pre-decoder. Note that the modified serial min-sum decoder executes only one iteration in a serial manner, there is no need to store $2N$ $L(q_{nm})$ values for the next iteration. Since the parity check matrix of SFCC-N4 code is periodic, then the additional memory for TSHD are $2J$ and L units corresponding to $L(r_{mn})$ and $L(q_{nm})$, respectively. For the central node, the additional memory cost is negligible.

III. Complexity Analysis

Since the parity check matrix of convolutional code is periodic, and the odd column weight J_1 and the even column weight J_2 of $\mathbf{H}$ satisfy $J = \frac{J_1+J_2}{2}$. The rate-1/2 SFCC-N4 code can be viewed as quasi-regular LDPC code with constant row weight $2J$ and average column weight $J = \frac{w+1}{2}$. As [14] indicates that the minimum values $\min_{n' \in N(m)\setminus\{n\}}\{\beta_{n'm}\}$ associated with each check sum m require $2J + \lceil \log_2 2J \rceil - 2$ comparisons with the help of a binary tree. Compared with the parallel min-sum algorithm, the additional computation for the modified serial min-sum decoder is caused by the updating rule. Identifying the two minimum values among the three (γ_1, γ_2 and γ_u) for each check m needs at most $4J$ comparisons. According to [14], the complexity of the modified serial min-sum decoder is shown in Table I.

TABLE I
Decoding Complexity of the Modified Serial Min-Sum Decoder for $2N$ Bits

Operation	Number of Computations
$\min_{n' \in N(m)\setminus\{n\}} \beta_{n'm}$	$N(6J + \lceil \log_2 2J \rceil - 2)$
$L(q_n)$ and $L(q_{nm})$	$4NJ$

Each time, M-algorithm operates mainly in two parts: (1) The path extension unit extends M_p survivor paths to $2M_p$ paths and calculates the associated metrics. (2) The path selection unit sorts the $2M_p$ paths and selects the M_p survivor paths based on the metrics. The sorting operation requires $2M_p \log(2M_p)$ comparisons using the fast sorting algorithm. Note that we only need to find out the M_p largest values among the $2M_p$ metrics. In [15], an improved technique is introduced, which provides $3M_p - 1$ comparisons for sorting $2M_p$ metrics per trellis stage. The complexity of M-algorithm is summarized in Table II.

TABLE II
Decoding Complexity of M-algorithm for N Trellis Stages

Operation	Number of Computations
path extension	$4NM_p$
path selection	$N(3M_p - 1)$

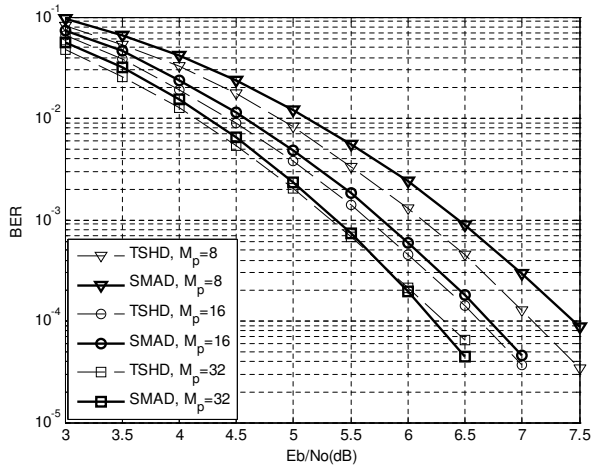
The overall computations for TSHD and SMAD are $N(10J + \lceil \log_2 2J \rceil - 2) + \theta(7M_p - 1)$ and $N(7M_p - 1)$, respectively, where θ represents the number of trellis stages re-decoded by M-algorithm in adaptive list decoder. Statistically speaking, the ratio $\rho = \frac{\theta}{N}$ is asymptotically a constant value as N increases, where $\rho \leq 1$. Then the computations for TSHD and SMAD per trellis stage are $10J + \lceil \log_2 2J \rceil - 2 + \rho(7M_p - 1)$ and $7M_p - 1$, respectively.

IV. Simulation Results

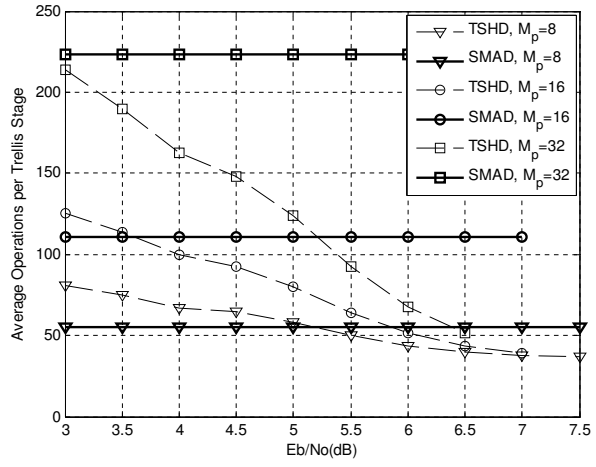
Simulations are performed to compare proposed scheme with conventional scheme using rate-1/2, length 200 code. In the conventional scheme, SMAD is used for decoding the optimum distance profile (ODP) code [8] with generator $G_o = (400000, 671166)_{octal}$, while in the proposed scheme, TSHD is applied to SFCC-N4 code with generator $G_4 = (400000, 620241)_{octal}$ for the same constraint length. As [16] suggested, the adoption of an error control code with simple decoding structures is quite energy efficient. Thus the list size M_p with small-to-medium value is used in our simulation.

Fig. 3 (a) compares the bit error rate (BER) performance of two schemes with hard decision decoding. At the BER of 10^{-4}, we could see that TSHD is about 0.4dB superior to SMAD with $M_p = 8$, and the performance narrowed as M_p increases. For $M_p = 32$, the two schemes almost have the same performance. The fact is that TSHD mitigates the path loss problem for small M_p with the help of the modified serial min-sum decoder.

Fig. 3 (b) depicts the complexity comparison mentioned in Fig. 3 (a). TSHD achieves the significant computation reduction to SMAD in medium-to-high SNR region. It should be emphasized that TSHD is less efficient in the low SNR



(a)



(b)

Fig. 3. (a) BER performance of TSHD and SMAD. (b) The average complexity of TSHD and SMAD in (a).

region with small list size. However, this can be avoided by sending the received data to adaptive list decoder directly, the first stage of TSHD is not used. Bypassing the first stage means the complexity of TSHD is the same with that of SMAD. At BER 10^{-4}, TSHD is about 0.4dB better than SMAD, and the saving is 33.4% for $M_p = 8$. In addition, TSHD has almost the same performance to SMAD, and the saving is up to 74.6% for $M_p = 32$.

The conversion between decoding delay and complexity is shown in Table III. According to Fig. 3 (a), the decoding delay effect on the BER performance and the ratio ρ is tested at 6.5dB with $M_p = 8$. It can be seen that $d_L > 3L$ has no performance improvement for TSHD, but the complexity increase as ρ grows.

TABLE III
THE EFFECTS OF DIFFERENT DECODING DELAY FOR TSHD

d_L	0	L	$2L$	$3L$	$4L$	$5L$
BER	1.1245e-002	4.9360e-004	4.4447e-004	4.5688e-004	4.4985e-004	4.5665e-004
ρ	2.3097e-002	4.4484e-002	5.9514e-002	6.9713e-002	7.7428e-002	8.2215e-002

V. CONCLUSIONS

In this paper, an FEC scheme was presented for the trade-off among decoding delay, complexity, and performance according to user's needs. The proposed scheme combining two decoders without interleaver has at least three merits:
1) Adaptive optimization;
2) Low average computation without performance loss;
3) Suitable for short packet transmissions.

The adaptive list decoder is superior to the single M-algorithm decoder with small survivor paths. Compared with SMAD in medium-to-high SNR region, the ALD has noticeable computational savings by hard decision decoding without performance loss. Moreover, the adaptive optimization mechanism makes TSHD suitable for WSNs applications.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] S. Howard, K. Iniewski, and C. Schlegel, "Error control coding in low-power wireless sensor networks: When is ECC energy-efficient?" *EURASIP Journal on Wireless Communications and Networking*, pp. 1–14, Apr. 2006.

[3] J. A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile, "IEEE 802.15.4: A developing standard for low-power, low-cost wireless personal area networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 15, no. 5, pp. 12–19, Sep. 2001.

[4] E. shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," *ACM SIGMOBILE'01*, pp. 272–287, July 2001.

[5] Y. Sankarasubramaniam, I. F. Akyildiz, and S. W. McLaughlin, "Energy efficiency based packet size optimization in wireless sensor networks," *IEEE International Workshop on Sensor Network Protocols and Applications, 2003*, pp. 1–8, 2003.

[6] M. R. Islam and K. Jinsang, "Capacity and BER analysis for Nakagamin channel in LDPC coded wireless sensor network," *International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2008*, pp. 167–172, 2008.

[7] K. S. Zigangirov and V. D. Kolesnik, "List decoding of trellis codes," *Probl. Contr. Inform. Theory*, pp. 347–364, 1980.

[8] H. Osthoff, J. B. Anderson, R. Johannesson, and C. F. Lin, "Systematic feed-forward convolutional encoders are better than other encoders with an M-algorithm decoder," *IEEE Trans. Infor. Theory*, vol. 44, no. 2, pp. 831–838, Mar. 1998.

[9] A. J. Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Infor. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[10] Y. He, D. Haccoun, and C. Cardinal, "Performance comparison of iterative BP and threshold decoding for convolutional self-doubly-orthogonal codes," *VTC '07- Spring*, pp. 2000–2004, 2007.

[11] A. W. Eckford and R. S. Adve, "A practical scheme for relaying in sensor networks using repeat-accumulate codes," *The 40th Annual Conf. on Information Sciences and Systems, 2006*, pp. 386–391, 2006.

[12] Y. Ye, X. Liu, and H. Cho, "An energy-efficient single-hop wireless sensor network using repeat-accumulate codes," *International Conference on Communications, Circuits and Systems, 2008*, pp. 419–423, 2008.

[13] F. Yang, Z. Luo, B. Tian, and Y. Li, "Design of turbo-like codes for short frames," *IEEE Commun. Lett.*, vol. 14, no. 2, pp. 172–174, Feb. 2010.

[14] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May. 1999.

[15] A. D. Kot, "A linear-time method for contender sifting in breadth-first decoding of error control codes," *Canadian Conference on Electrical and Computer Engineering, 1993*, vol. 1, pp. 262–265, 1993.

[16] S. Kasnavi, S. Kilambi, B. Crowley, K. Iniewski, and B. Kaminska, "Application of error control codes (ECC) in ultra-low power RF transceivers," *Architecture, Circuits and Implementtation of SOCs, 2005*, pp. 195–198, Sep. 2005.